AppleScript Help

Essentials

About AppleScript

As you use your Mac, you'll find some tasks that you do over and over. With AppleScript you can write scripts and applications that can automate repetitive tasks.

AppleScript is a powerful and versatile scripting language that is built into OS X. You can use AppleScript to create shortcuts, automate repetitive tasks, or even make custom applications that save you a great amount of time. AppleScript lets you control scriptable applications.

An AppleScript script consists of English-like commands that perform actions on objects. Some examples of actions are open, close, or get the name of. Some examples of an object are an application, a window, or a preference setting.

A script can control one or more applications. To be scriptable, an application must define the terms you can use in your scripts. OS X comes with several scriptable applications.

Scripts can also control applications over a network, so you can automate tasks on more than one computer.

In fact, you have been using AppleScript without knowing it. Many functions in the system and in applications are really AppleScript scripts that are called from within an application. For example, in this help system, whenever you click a link that opens an application or a file, it is an AppleScript script that performs the action.

RELATED TOPICS

How AppleScript works
About AppleScript Editor
About the AppleScript language
AppleScript resources on the Internet
AppleScript Studio development environment

Essentials

About AppleScript Editor

AppleScript Editor is an easy-to-use scripting application that can read, write, record, and save AppleScript scripts.

Here are some of the AppleScript Editor features:

- Provides a unified, easier-to-use interface
- Lets you find and replace in text in scripts
- Does automatic wrapping of long script statements
- Lets you do multiple undos
- Provides shortcut menu scripts for faster script writing
- Supports application services

- Lets you access application scripts by using the Library Palette
- Provides a Script Assistant to help you complete statements
- Provides the result history and event log history so you can examine what is going on in a script

RELATED TOPICS

About the AppleScript language
AppleScript Studio development environment

Essentials

Create AppleScript scripts

You don't need to know anything about programming to benefit from AppleScript. In fact, you don't need to write one word of script. There are hundreds of useful AppleScript scripts included with OS X that you can begin using, and thousands more are available on the web.

If you wish to go a bit deeper, you can take existing scripts and suit them to your needs with minor modifications.

Of course, you will gain the most advantage by creating your own scripts. Learning how to write scripts will enable you to create useful tools and time-saving automations that will make your work more productive.

You can modify or create your own scripts with AppleScript Editor, which provides a simple environment in which to write your scripts. All of the examples in AppleScript Help were done using AppleScript Editor.

- Open AppleScript Editor.

  Type the following text into the AppleScript Editor window.

  ```
  say "Welcome to AppleScript!"
  ```

  Click run.

RELATED TOPICS

About AppleScript Editor
AppleScript Studio development environment

Essentials

How AppleScript works

AppleScript works by sending messages, called “Apple events.” When you run a script, AppleScript sends Apple events to the system or applications that carry out the commands.

In many cases, applications or the system will return a result to AppleScript. For example, if a script sends a request to an application to get the name of an object, the application returns the name. If the command cannot be executed, the application or AppleScript may return an error message. You can see the results returned in AppleScript Editor.

In general, you don't need to be aware of the mechanics of Apple events. AppleScript does all of this in the background. You just need to write and run your scripts.

RELATED TOPICS

About AppleScript
About the AppleScript language

Essentials

About applets

A script that is saved as an application is called an AppleScript application, or applet. Applets run automatically when opened and perform the commands in the script. Applets can be small functions containing only a couple of commands, or complex applications containing thousands of lines.

Applets can be saved as editable or run-only. An editable applet still contains the text of the script, so it can be edited and recompiled at any time. To edit or change an applet, drag it onto the AppleScript Editor icon. When you save a script as run-only, the editable text portion is removed and the file can no longer be edited or recompiled.

To edit an applet that was saved as editable, you open it in AppleScript Editor or another AppleScript editing application. Applets require an “on run” handler, which begins running the script when the applet is opened.

- This example shows an “on run” handler.

```
on run
        --first command
        --second command
end run
```

RELATED TOPICS

About droplets
AppleScript file types

Essentials

About compiled scripts

A compiled script is an AppleScript file in an editable form that you can read and edit in AppleScript Editor, or in another AppleScript editing application.

When you open a compiled script, it appears in the AppleScript Editor but does not automatically perform any of the commands. This allows you to make any changes to the script without having it actually run. You can then compile the script, which reads and converts all the text into AppleScript code within the file. The compiled script can be run. A compiled script has a file extension of .scpt.

When you save the file as a compiled script, the text still remains in the file along with the code. You can therefore edit the compiled script again at any time.

RELATED TOPICS

About applets
About droplets
AppleScript file types

About droplets

A droplet is an AppleScript script that runs when you drag files or folders onto its icon. The script performs actions on the files or folders.

Just as with applets, a droplet can be saved as editable or run-only. An editable droplet still contains the text of the script, so it can be edited and recompiled at any time. To edit or change a droplet, drag it onto the Script Editor icon. When you save a script as run-only, the editable text portion is removed and the file can no longer be edited or recompiled.

Droplets require an “on open” handler, which begins running the script when the droplet is opened. The “on open” handler has a parameter that contains a list of the items dragged to the droplet. Part of the droplet script examines the list, and another part acts on the items. Unless the droplet is designed to act on only a single item, it should include a repeat loop that iterates through the items in the list, performing actions on each item in turn.

- Here is an example of a droplet handler with the “on open” and parameter list.

```
on open (theItemsDropped)
        --first command
        --second command
end open
```

RELATED TOPICS

About applets
About compiled scripts
AppleScript file types

Use AppleScript dictionaries

To find out which commands and objects you can use to control an application, you can look at that application’s dictionary. A dictionary contains a set of terms that are understood by an application.

You can view an application’s dictionary in AppleScript Editor by selecting the application in the Library window, or by dragging the application’s icon onto the AppleScript Editor icon.

Terms in a dictionary are organized into groups of related terms called “suites.” Commands and objects are in bold text.

1. In AppleScript Editor, choose Window > Library.
2. Double-click the name of the application in the Library window.
3. Select a suite.

   The terms in the suite and their descriptions are displayed.

The AppleScript language is arranged into different categories:

- Suites: Groups of related terms. In the AppleScript Editor Dictionary, suites appear in the first column, designated by an orange letter “S.”
- Commands: request an action or a result. Commands, which are also referred to as “events,” appear in the second column, designated by a blue letter “C,” (also referred to as an event.)
- Classes: Scriptable objects. Class descriptions list the properties and elements that exist in the class. Classes appear in the second column, designated by a lavender letter “C.”
- Properties: Objects that contain values. Properties appear in the third column, designated by a lavender letter “P.”
- Elements: Objects contained in other objects. For example, a paragraph may contain many words. Elements can appear in the third column, designated by an orange letter “E.”

RELATED TOPICS

Class definitions
Command definitions
Use AppleScript to command applications

OPEN

AppleScript Language Guide: Commands Reference
AppleScript Language Guide: Class Reference

Essentials

Solve Problems

If you’re having a problem using AppleScript, explore the following topics for information to solve the problem.

If an error appears when a user clicks Cancel in a Choose From List dialog

If your folder action is triggered unexpectedly

If your script doesn't recognize a disk

Basic commands

Class definitions

A class definition describes the properties and elements of a scriptable object. A class can be compared to a noun in a sentence, and the class definition is like a noun’s dictionary definition.

Typically, a class definition in an AppleScript dictionary begins with the term, in bold, followed by a short description of the term on the same line.

This is followed by a list of any elements or objects that belong to or constitute the scriptable object. An elements list is not always included in all class definitions because some scriptable objects contain other scriptable objects and some do not.

The class definition ends with a list of any properties or qualities of the scriptable object and their values.

A property entry begins with the name of the property, followed by:

- A description of the data format in which the property’s value is stored, such as integer (number), point (a list of two numbers), string (textual data), and Boolean (true or false)

- An optional type indicator, [r/o], that indicates that the value of the property is "read-only" (it can only be read and not changed)
- A short description of the property

If an object has a plural form, you can refer to more than one object.

If an object has a list of elements, it contains other objects. The element name, which you can use in your script, appears in bold. The plain text describes how to refer to the element.

Objects can have properties. Property names appear in bold. The plain text describes the type of value the property contains, such as a string. Properties can be a constant value or one of a list of constants. A script can get the value of a property and can set the value of a property, unless the property has [r/o] (read-only) after the syntax.

RELATED TOPICS

Command definitions
Object definitions
AppleScript commands
Use AppleScript dictionaries

OPEN

AppleScript Language Guide: Class Reference

Basic commands

Command definitions

Commands are terms in AppleScript that perform actions. Commands in an AppleScript dictionary are displayed in the center column and designated by a blue letter "C". The definition of a command includes a brief description of what the command does and its syntax. The syntax is the way you write the command in a script. The syntax may include several parts.

Bold words are the terms you use in a statement. You must write these terms exactly as they appear in the dictionary.

Blue text words indicate information you provide. For instance, you might identify an object or a variable you have defined.

Commands sometimes have optional parts, which appear in brackets. You do not have to use these in a statement.

RELATED TOPICS

Class definitions
Use AppleScript dictionaries
Event definitions
Object definitions

OPEN

AppleScript Language Guide: Commands Reference

Basic commands

Event definitions

An event definition describes a verb or command used to target an action at a scriptable object. Events are the actions or

verbs that objects can accomplish or participate in.

Typically, an event definition begins with the term followed by a short description of the term on the same line.

This is followed by a list of any parameters used in the execution of the command. Optional parameters are enclosed in brackets; required parameters are not.

Each parameter entry begins with the name of the parameter followed by a description of the data format used with the parameter, and ends with a short description of the parameter.

The definition of a command includes a brief description (the underlined text) of what the command does and its syntax. The syntax is the way you write the command in a script. The syntax may include several parts.

Bold words are the terms you use in a statement. You must write these terms exactly as they appear in the dictionary.

Plain text words indicate information you provide. For instance, you might identify an object or a variable you have defined.

Commands sometimes have optional parts, which appear in brackets. You do not have to use these in a statement.

Commands act on objects. The definition of an object shows the syntax for identifying the object, including its name and a brief description (the underlined text). The class of an object determines certain object characteristics, such as the commands you can use with the object.

RELATED TOPICS

Database events
Class definitions
Command definitions
Object definitions
How AppleScript works

Basic commands

Object definitions

Commands act on objects. The definition of an object in an AppleScript dictionary shows the syntax for identifying the object, including its name and a brief description (the underlined text). The class of an object determines certain object characteristics, such as the commands you can use with the object.

If an object has a plural form, you can refer to more than one object.

If an object has a list of elements, it contains other objects. The element name, which you can use in your script, appears in bold. The plain text describes how to refer to the element.

Objects can have properties. Property names appear in blue bold. The plain blue text describes the type of value the property contains, such as a string. Properties can be a constant value or one of a list of constants. A script can get the value of a property and can set the value of a property, unless the property has “[r/o]” (read-only) after the syntax.

RELATED TOPICS

Use AppleScript to command applications
Command definitions
Class definitions
Tell an object what to do
Use AppleScript dictionaries

OPEN

AppleScript Language Guide: Class Reference
AppleScript Language Guide: Properties

Basic commands

If statements

An if statement is a command that is executed only if certain conditions are met. Each if statement contains a Boolean expression that is evaluated as either true or false. The commands contained within the if statement are executed only if the value of the Boolean expression is true. The if statement ends with an end if. For example:

```
if amountReceived > 100 then
        set memberStatus to "special"
end if
```

In the above statement, if the Boolean expression is true (the amount received is greater than 100), then the commands within the if statement are executed. If the Boolean expression is false, the commands are not executed.

The if statement can also be constructed with an else statement. This combination is known as an if-then-else statement. For Example:

```
if amountReceived > 100 then
        set memberStatus to "special"
else
        set memberStatus to "general"
end if
```

In the above statement, if the first Boolean expression is true (the amount received is greater than 100), then the commands in the first part of the statement are executed. If the first Boolean expression is false, the commands in the second part of the statement following else are executed.

You can nest if statements within each other to handle multiple conditions.

To learn more about if statements in AppleScript, see the AppleScript Language Guide.

RELATED TOPICS

Repeat statements
Command definitions
Set variables in AppleScript
Constant values

OPEN

AppleScript Language Guide - If

Basic commands

Repeat statements

You can use repeat statements to perform tasks multiple times depending on conditions. A repeat statement continues to

run a group of statements in a loop until a condition is met and the loop is exited. You can nest repeat statements within each other to respond to varying conditions.

AppleScript has a range of repeat statements that can respond to different conditions.

repeat (forever)

Repeats continually in a loop. Use an exit statement to leave the loop.

```
repeat
	-- statement
	-- statement
	if (x = 5) then exit repeat
end repeat
```

repeat (number) times

Repeats for the specified number of times.

```
repeat 5 times
	-- statement
	-- statement
end repeat
```

repeat while

Repeats while a specified condition is true.

```
set x to 1
repeat while (x < 5)
	-- statement
	-- statement
	set x to (x + 1)
end repeat
```

repeat until

Repeats until a specified condition is true.

```
set x to 1
repeat until (x = 5)
	-- statement
	-- statement
	set x to (x + 1)
end repeat
```

repeat with (variable) from (startValue) to (stopValue)

Sets the variable to the startValue and then increments the variable after each time through the loop. The repeat ends when the variable reaches the stopValue.

```
repeat with x from (5) to (25)
        -- statement
        -- statement
end repeat
```

repeat with (variable) in (list)

Sets the variable to the first item in the list. With each pass through the loop the variable is set to the next item. The repeat ends when all the items of the list have been used.

```
set myList to {234, 342, 681, 874}
repeat with x in myList
        -- statement
        -- statement
end repeat
```

To learn more about repeat statements in AppleScript, see the AppleScript Language Guide.

RELATED TOPICS

If statements
Set variables in AppleScript
Constant values

OPEN

AppleScript Language Guide - repeat

Basic commands

Constant values

A constant is a reserved word in AppleScript that contains a predetermined value. AppleScript contains numerous constants that you can use to get information about objects, perform comparisons, or do arithmetic calculations. Applications and other objects can define their own constants. These constants are defined in the value class definitions.

To see the value of a constant, paste the following example into AppleScript Editor, and then click Run. Click Result to view the value of the constant.

The result is also a constant. It is a script constant that refers to the value returned by a command.

```
get the current date
```

Here are some other examples of constants:

- Arithmetic constants: pi
- Boolean constants: true, false
- Date and time constants: minutes, hours, days, weeks
- String constants: return, space, tab
- Script constants: anything, current application, it, me, missing value, my, result

To learn more about constants in AppleScript, see the AppleScript Language Guide.

RELATED TOPICS

Set variables in AppleScript
Object definitions
Class definitions
Event definitions

OPEN

AppleScript Language Guide - constants

Basic commands

Set variables in AppleScript

A variable is a container created within a script that is used to store a value. You specify a name for the variable, also called an identifier, when you create it. When a script is run, you can place a value into the variable, access the value, or modify it. Variables last only while the script runs. When the script closes, the value is lost.

- These examples show how to specify a unique identifier and assign a value to it.

```
set myVar to "153"

set x to "Quarterly Reports"

set x to false
```

To see how values are stored in a variable, paste these examples into AppleScript Editor, and then click Run. Click Result to view the value stored in the variable.

To learn more about variables in AppleScript, see the AppleScript Language Guide.

RELATED TOPICS

Constant values
If statements
Repeat statements

OPEN

AppleScript Language Guide - variables

Basic commands

Specify an application's location

Applications in OS X are bundled as "packages." A package is a special directory that contains the folders and files that make up the application. Because of this, the "path to" and "info for" commands may return different results for an application.

If you use the "path to" command for an application, the path usually ends with ".app." File extensions might not be

displayed in the Finder as part of the filename, but you can still use extensions to identify a file in a script.

If you use the “info for” command to get information about an application, the Folder property of the result is usually “true.” You can get more information about the file and its name by using additional properties of the result, such as extension hidden, name extension, or displayed name.

RELATED TOPICS

path to
info for
Refer to folders, files, and disks
choose application
Finder scripting commands

Finder commands

Refer to folders, files, and disks

AppleScript allows you to refer to certain folders, files, and disks by name, without including a path. AppleScript can identify the location of such items as the Documents folder, desktop, home folder, and startup disk.

- This example shows how to refer to a special folder by using its name in a script.

```
set x to the path to the desktop folder
```

RELATED TOPICS

Refer to the Finder in scripts
path to
info for
Finder scripting commands

Finder commands

Refer to the Finder in scripts

To use Finder commands, you direct the commands to the Finder in a tell statement, just as you would direct commands to an application.

- Here is an example using the tell statement in a script. The tell statement directs the number of windows function to “Finder.app.”

```
tell application "Finder.app"
        return the number of windows
end tell
```

RELATED TOPICS

Finder
Tell an object what to do

Refer to folders, files, and disks
Object definitions
Finder scripting commands

OPEN

AppleScript Language Guide - Object specifiers

Finder scripting commands

Here are some commands you can use with the Finder:

activate

alias file

alias list

application file

clean up

close

column

Computer-object

container

copy

count

data size

delete

desktop window

desktop

disk

document file

duplicate

eject

empty trash

exists

file

file information

find

Finder

Finder window

folder

get

icon view options

information window

Internet location file

item

list view options

make

move

moving folder window for

open

package

path to

preferences

preferences window

print

quit

Refer to folders, files, and disks

Refer to the Finder in scripts

reveal

select

set

sort

trash

update

window

Standard scripting addition commands

The scripting additions adds many useful commands that you can include in your scripts.

adding folder items to

beep

choose application

choose color

choose file

choose file name

choose folder

choose from list

choose URL

clipboard info

close access

closing folder window for

current date

delay

dialog reply

display

display dialog

do shell script

download

FTP item

get eof

info for

list disks

list folder

load script

mount volume

offset

open for access

open location

opening folder

path to

POSIX path

random number

read

round

run script

say

scripting components

set eof

set the clipboard to

set volume

store script

summarize

system attribute

time to GMT

upload

URL

write